Introduction to Python Programming

Learning Outcomes

By the end of this session, you will have covered the following learning outcomes:

- I can use the **print()** function to display the output of a python program.
- I can create **variables** to store data in a python program.
- I can write **comments** to provide explanations of how code works.
- I can write **arithmetic operators** to perform mathematical operations.
- I can **determine** the data types of different types of data.
- I can use **lists**, **dictionaries,** and **sets** to store data.
- I can use **conditional (if)** statements to check for conditions in data.
- I can use **for loops** and **while loops** to iterate through data.
- I can create **functions, classes, and objects** to write reusable code.

Overview

The **Python programming language** is one of the most popular programming languages in Data Science. A comparison from a Kaggle survey [Link] of several programming languages used in data science shows that the programming language continues to grow in popularity due to ease of use, vast developer community, and appropriate data science libraries.

By the end of this session, you will be able to demonstrate the use of crucial python programming concepts such as printing, variables, comments, arithmetic operators, data types, lists, conditional (if) statements, dictionaries, sets, for loops, while loops, functions, classes, and objects.

Terminology

- **Comments** are used in our code to provide explanatory information about the code.
- **Variables** - These are names that we give to computer memory locations used to store values in a computer program.
- **Arithmetic Operators** - These are mathematical functions that take two values and perform a calculation on them.
- **Data types** - A data type is a classification that specifies which type of value a variable contains.
- **Lists** - A list is a compound data type that allows for storing items of different data types.
- **Conditional (if) Statements** - We use conditional-if statements to make decisions. For example, to run code only when statements are true.

- **Dictionaries** - We use a dictionary to store key-value pairs. The retrieval, adding, removing, and modifying of values happens through keys.
- **Sets** - A set is an unordered collection of elements with no duplicate elements. Set elements can be of any data type but cannot have mutable elements like lists, sets, or dictionaries.
- **For loops** are used to iterate over a sequence, i.e., list, dictionary, string, etc.
- **While loops** execute a set of statements as long as a condition is true.
- **Functions** - A function is a block of organized, reusable code.

Main Resources

Reading

Printing

- print() is one of the most popular python commands.
- This command print means "show on the screen".

Variables

- Variables are storage containers for data.
- For example, a food variable could contain "Spaghetti" or "Rice" or any other value.
- The value of a variable can change depending on conditions or information passed to the program.
- Variable names must begin with a letter, underscore, or non-number character. Each programming language has its naming conventions to guide how variables are named.
- Every programming language has reserved words. We can't use these words as variable names, for example, *Date*. Instead, we might name your date-related variables *dte* or *StartDate*.
- Once we create variables, we are then able to perform operations taking into account the stored values.

Comments

- We use comments to explain how a program works.
- Comments don't have any effect on your program.
- Comments have the following goals:
    - To explain what a particular code does.
    - To explain something which might not be evident to the reader.
    - To clarify your intention behind a particular line or block of code.
    - To serve as a reminder to change something in the future.

Arithmetic Operators

- Arithmetic operators perform mathematical operations such as addition, subtraction, multiplication, division, modulus, exponentiation, etc.

- The addition operator will produce the sum of numeric operands or string concatenation.
- The subtraction operator will subtract two operands and produce their difference.
- The division operator will produce the quotient of its operands where the left operand is the dividend, and the right operand is the divisor.
- The multiplication operator will produce the product of the operands.
- The modulus operator returns the remainder when we divide an operand by a second operand, i.e., 5 / 2.
- The exponentiation operator will return the result of raising the first operand to the power second operand.
- We use the increment operator to increment (add one) to its operand and return a value.
- The decrement operator will decrement (subtract one from) its operand and return a value.

Data Types

- A data type is a classification that allows specifying which type of value a variable contains.
- It also allows us to determine what type of mathematical operations we can apply to any given data.
- There are different types of python data types: integer, float, string, lists, dictionaries, etc.
- The integer value is a positive or negative whole number.
- The float value is a Real number with floating-point representation.
- The string value is a sequence of characters.
- A list can contain data of different data types.
- A dictionary is an ordered set of a key-value pair of items.
- Other data types include tuples and sets.

Lists

- A list is a compound data type that allows for the storage of items of different data types.
- Lists are mutable, which means that we can change an item in the lists by accessing it directly.
- Below is an example of a list:

  cakes = ['blueberry', 'vanilla', 'strawberry', 'chocolate', 'apple'].
- Each element or value that is inside a list is called an item.
- We work with lists when we would like to work with many related values, i.e., food, songs, emails, etc.
- The various operations that we can perform on lists are indexing (which allows us to access its items), slicing, updating, appending and removing.

Conditional (if) Statements

- We use conditional if statements to make decisions.
- We write an "if statement" using the if keyword and then print out the result when one of the conditions is true or false.

- The "elif statement" is used when we have a third possibility as the outcome. We can use multiple elif conditions to check for the 4th, 5th, 6th possibilities in our code.
- We can also nest our if statements by including if statements within our if statements.
- While working with if statements, we also have to take into consideration indentation. Indentation is the white space at the beginning of a line that python.

Dictionaries

- A dictionary is a data type that stores key-value pairs. It enables us to retrieve, add, remove, modify values using keys.
- To create a dictionary, we place items inside curly braces separated by a comma, as shown in the example below. These items usually have a key and value.
- dictionary_example = {1: 'apple', 2: 'ball'}
- While working with dictionaries, we can retrieve, modify, add and remove elements from the dictionary. Thus, having them as mutable.
- Dictionary key-value pairs are unordered, meaning that the added key-value pairs don't necessarily reflect what order they will have when they are retrieved.
- They can also be nested, which means that a dictionary can contain another dictionary.

Sets

- A set is an unordered collection of elements with no duplicate elements.
- Set elements can be of any data type but cannot have mutable elements like lists, sets, or dictionaries as elements.

For Loops

- We use a for loop to iterate over a sequence, i.e., list, dictionary, string, etc.
- The loop continues until we have gone through the final item list.
- We separate the body of the for loop from the rest of the code using indentation.
- We can also have for loops within for loops; This is what we would call nesting.
- We can use a break statement in a for loop to stop the loop before it has looped through all the items.

While Loops

- A while loop executes a set of statements as long as a condition is true.
- A while loop requires we initialize a variable outside the loop.
- The break statement within the while loop would immediately terminate the loop entirely then proceed to the first statement following the loop body.
- The continue statement would end the current loop iteration and jump to the top of the loop back to the control expression to determine whether the loop should execute again or terminate.

Functions

- A function is a block of organized, reusable code.
- A common usage of functions is to implement mathematical functions.
- As we will get to see, a function block begins with the keyword *def* followed by the function name and parentheses *()*.

- We can define a function by giving it a name, specifying the parameters to be included in the function then including blocks of code within the function's body.
- We use parameters to pass input data to functions and, as a result, get an output.
- We use the return keyword to exit a function and pass back some value from the function.
- Lastly, to execute a function, we call the function name.

Classes and Objects

- A class is a user-defined blueprint that performs a set of related tasks.
- Classes contain variables and methods which can be accessed and used by creating an instance of that class (i.e., an object).
- Attributes are the variables that belong to a class. Methods are functions defined as part of a class.
- The __init__() function is used to perform operations when creating an object, i.e., assign values to object properties, which are the mentioned attributes.
- A `self` parameter refers to the current instance of the class and is used to access variables that belong to the class.

Further Resources

- Python Basics for Data Science. [Link]
- Python Documentation. [Link]